IMPLEMENTING A GENERIC COMPONENT-BASED FRAMEWORK FOR TELE-CONTROL APPLICATIONS

Avraam N. Chimaris, George A. Papadopoulos
Department of Computer Science, University of Cyprus
75 Kallipoleos Street, POB 20537, CY-1678, Nicosia, Cyprus
Email: chimaris@cytanet.com.cy, george@cs.ucy.ac.cy

Keywords: Distributed Applications, Component-Based Systems, Generic Frameworks, XML/XSL, TCP/IP.

Abstract: The rapid growth of distributed systems is one of the major facts in today's network-oriented community. The implementation of generic frameworks, consisting of reusable components that can be used for the development of such systems is a necessity. There is a plethora of applications that can be developed in a distributed environment, such as audio/video tele-conferencing, groupware and collaborative computing environments, remote controlled services, etc. In this paper we design and implement a generic framework of components that can be used for the realization of Tele-Control applications. This category of applications focuses particularly on the issues of managing distributed units on remote end-systems. Such applications contain remote and administrative units that are connected and exchange data and control messages. In the analysis of our framework, we used UML for the specification, analysis and presentation of system operations. The distributed units of our framework are using XML messages and TCP channels for exchanging data and control messages. We implement a communication "protocol" that contains the basic messages that can be used in Tele-Control Systems. Finally, we are presenting two different applications, which are implemented by reusing the generic components of our framework.

1 INTRODUCTION

Distributed Systems are being extensively used in many parts of the industrial sector. There exist many categories of distributed applications, most of them with different types of requirements and operational functions. Although in general it is impossible to implement a generic framework for all of them, we would like to have available more focused frameworks for specific domains of distributed applications, comprising reusable components which can then be used to "instantiate" specific applications. In this paper we try to analyse the basic concepts of Tele-Control systems; most of the Tele-Control applications focus on the "controlling of remote units". This is an easy concept to understand, but to implement it requires a complex environment that contains middleware environments, predefined commands, structured data, communication channels, etc. Instances of a Tele-Control environment give rise to many modern applications, such as Tele-Medicine Systems, Domestic Appliances Control, Remote Observation Units, etc. In this paper we will generalise the model of a Tele-Control environment, while at the same time address all the common issues that are involved in such applications. In the following sections we will try to cover briefly the aspects of the Tele-Control framework we implemented and the constituent parts of such systems. We use both XML/XSL for formatting Tele-Control messages and TCP/IP channels for forwarding messages between units. The generic framework was implemented using Microsoft COM Technology (Szyperski, 1998), and towards the end of the paper we show how we can use it to develop two specific Tele-Control applications.

In the next section we analyse our framework by describing the communication issues and the constructing blocks of the system. In section 3 we present the two applications that we implemented by using our generic framework. We show some screenshots of the systems and we mention how these applications use the generic components to realize their requirements.

2 FRAMEWORK ANALYSIS

Any distributed application - homogenous or heterogeneous - communicates, exchanges and manages data via the network. Our generic framework contains four main categories of reusable components: Remote Units, Control Centre, Administration Units and Mobile Administrators.

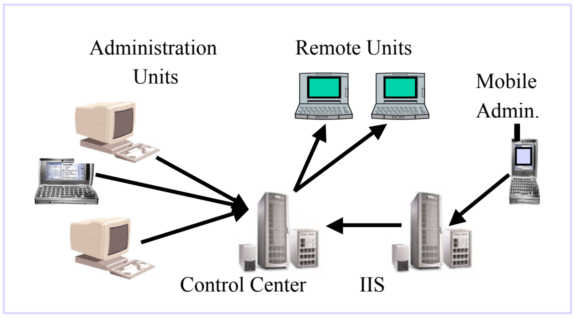


Figure 1: Tele-Control System

The Remote Units are small "intelligent" systems that have the ability to communicate with certain devices. The aim of our framework was to create the basic infrastructure of such systems and certainly not to create electronic mechanisms for communicating with external devices. In our example we are using components that are "simulating" the functionality of needed devices. The Control Centre is used in order to create the central communication unit in which both Remote Units and Administration Units can get connected. The Remote Units can send or receive messages by using their connection on the Control Centre. These messages can contain data about changes, the status of Units and control signals. The Administration Units, like the Remote Units, are able to send messages to Remote Units through Control Centre connections. The Administrators are connected either directly by TCP/IP channels or, in the case of Mobile Administrators, through the IIS (Internet Information Server) connection. The IIS Server is loading a "virtual" Administration space and it is constructing dynamic ASP (Active Service Pages) documents that are forwarded to the connected mobile devices. The Administration Units can get acknowledgments of changes on Remote Units by messages that are sent through the Control Centre connection. The Administration Units' messages are most of the times requests for changes, that update both Remote Units and copied data that is stored in other Units. All Units in the Tele-Control System (except WAP Administrators) can communicate and take part in peer-to-peer conference by using request/accept XML messages and Video/Audio streams. Figures 1 and 2 show the basic structure of a Tele-Control system and the Use Case diagrams (Figures 3 and 4) are the representation of the discrete set of work that will be performed by the users of the Tele-Control System. These users are handling the four categories of units we mentioned previously and they will be used as "actors" in the following Use Case diagrams. The Use Case diagrams are screening the major activities that users on Remote Units and on the Control Centre can perform. In Administrative Units and WAP Administrators the activities are quite similar. The users in these Units want to retrieve and update data of a selected Remote Unit by using their connection to the Control Centre. The difference between them is the fact that the Administrative Unit is directly connected to the Control Centre whereas the Mobile Administrator is using the IIS to perform its role.

Below we create certain sets of classes/ components for each Tele-Control Unit. These sets are constructing the fundamental functionality that resolved from role analysis as mentioned in the Use Case diagrams. The distributed sets of components are communicating through TCP channels and exchanging data based on their role in the system. In this section we are describing the general aspects that we followed in the analysis of the framework. Initially we will describe some communication

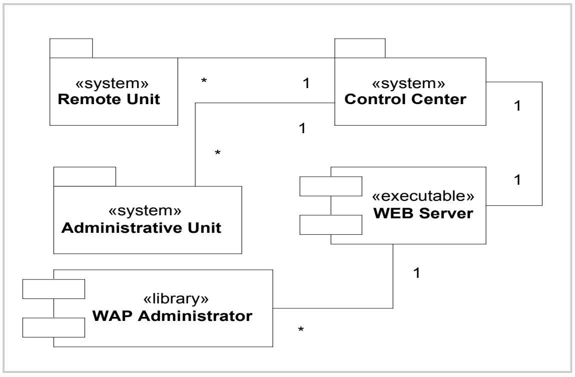


Figure 2: Tele-Control Components

issues, then we will show how XML/XSL messages are used, and finally we will extensively analyse the internal structure of each Tele-Control Unit.

2.1 Communication Issues

Any distributed system must necessarily have an infrastructure of communication channels between the remote units. In order to implement the communication aspect, it was necessary to create a communication "protocol" which could cover the needs in communication issues. We also created "clever" classes that are used for reading and parsing XML messages.

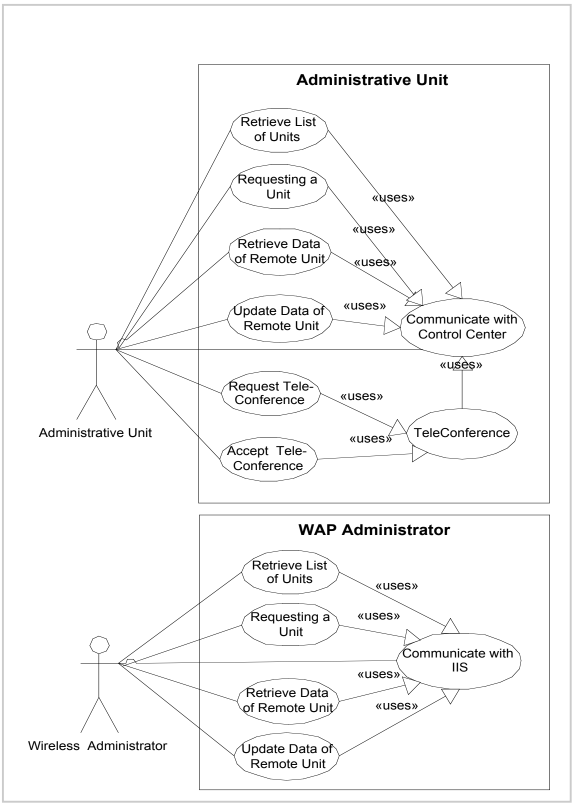


Figure 3: Use Case Diagrams

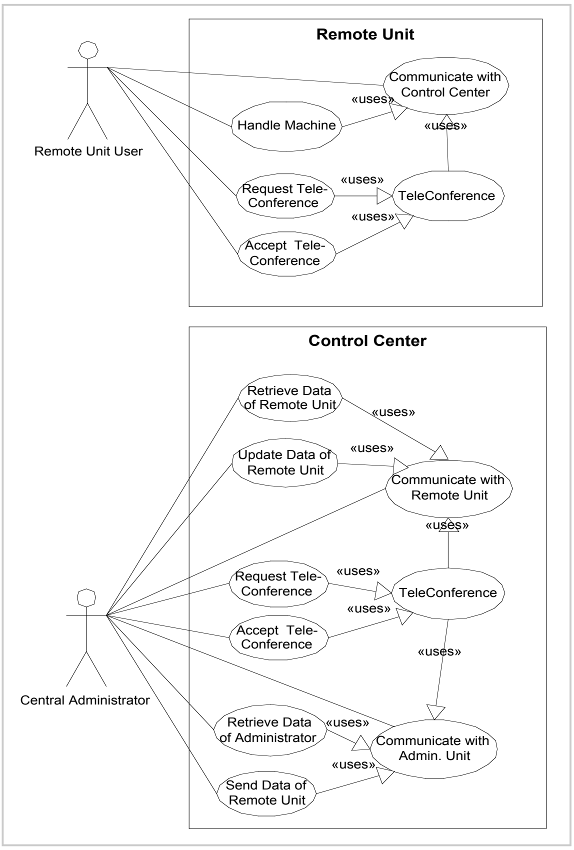


Figure 4: Use Case Diagrams

2.1.1 Communication messages

In the communication process use XML as an easy way of transmitting data. We created a DTD, which is constructed taking into account the needs of communication messages. Below we are showing the types of XML messages we are using:

XML Message: A Tele-Control unit uses this message in order to send an XML data structure to the connected peer.

XSL Message: This message is used in order to send an XSL Stylesheet to a connected peer. This Stylesheet is used to present the receipt XML data in the desired form on the connected peer unit.

Updating Message: This type of message is used in order to update XML data in the connected peer. This type of message is using XML "sub-tree" update information instead of sending the XML structure again.

Alert Message: This kind of message is used in cases of notifying Central Control Unit or Administrators of certain incidences on Remote Units.

General Message: This message is used in exchanging general messages between connected peers. General messages are separated by a certain element, which determines the specific type of the general message.

2.1.2 Communication classes

In our approach we tried to find a generic structure that could be used simply and efficiently in order to parse any XML structure. We implemented a class that can read and store an XML structure in a powerful tree, easily used by a programmer. This class, the clsTeleMsg, constitutes the main communication class of our communication. It is the mediator between the communicating parties and it is parsing messages and stores the XML data of the administrators and the distributed units. Generally speaking, this class has the capability to create recursively a tree of clsTeleMsg nodes, which hold the XML elements. We also implement classes that handle sockets (Communicators) with which the Tele-Control units send and receive the XML Messages. These classes are quite simple and they are initialising TCP channels on predefined ports that the Tele-Control System uses.

2.2 Tele-Control Units

In this session we describe briefly how Tele-Control units are implemented in order to support Tele-Control system functionality. The processes of such systems vary because they are built by using predefined sets of requirements. We tried to analyse the main requirements of such systems to determine the main functionality of the Tele-Control Units.

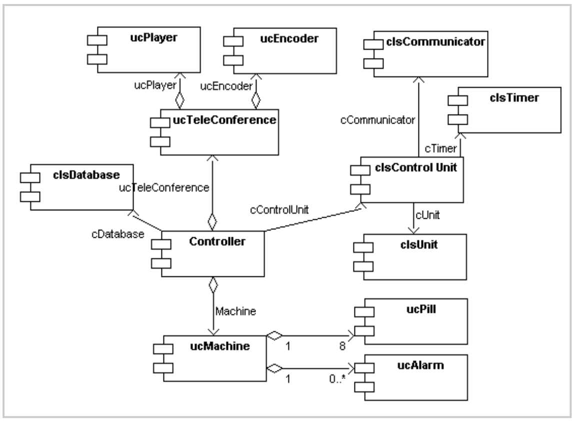


Figure 5: Remote Unit structure

2.2.1 Remote Units

The Remote Units are the units that are needed to be handled by the Administrative Units in the Tele-Control System. They have their data stored in an XML structure and they have functionalities for loading, updatting and communicating with other units. We have mentioned that Remote Units cannot be handled directly. They are connected with a Control Centre, which is responsible for retrieving and forwarding XML messages to the Remote Unit. The Remote Unit is using a clsUnit class for storing its data and also a clsCommunicator class for connecting to the Control Centre. In Figure 5 we are presenting a class diagram for the Remote Unit. In this diagram both clsUnit and clsCommunicator are shown; these are handled by a Controller Class. The controller classes are used in all Tele-Control Units. This type of class is using the parsed XML messages to execute a defined process on a local unit. For example, the Remote Units' conroller (clsControlUnit) can receive an update XML message and use it in order to update local XML data (clsUnit). Below we are summarizing some of the major activities that this controlling class can perform:

- Handling the Communicator for connecting to the Control Centre.
- Sending the initial data of a Remote Unit to the Control Centre. Data is loaded into a clsUnit class, which is contained in the Controller. The XSL Stylesheet is also stored and transmitted in the initial stage.
- Performing the activities on XML messages arrivals. These messages are mainly updating messages and teleconferencing messages.

The role of this category of units is to retrieve and update data on certain devices. These devices must be electronic machines that are connected on the Remote Units and handled by a defined set of commands. In this paper we will avoid defining any external signals because we put emphasis on how these Tele-Control units cooperate. In the two applications that we implemented by means of this framework, we used certain "simulators" that have the role of external machines. These "machine components" are components that encapsulate the basic functionality that they are expected to have, due to their role in the application we are implementing. In the analysis of the framework we will use the Tele-Medicine application, which is an application that handles medication of remote patients. In this application the Remote Units are machines that handle a set of stacks that hold a number of pills. A medication is submitted at the machine and only when the medication time is reached, the patient is notified in order to take his pills. So if we summarize the general components of the Remote Unit and the "machine" that is needed in order to create such a unit, we will have effectively generated the diagram in Figure 5. It is obvious that that a "programmer-defined" component is needed (Controller) in order to coordinate the four major parts of the Remote Unit, which are the following ones: Controller (clsControlUnit), Teleconference component (ucTeleconference), Machine component (ucMachine), and a Database class (clsDatabase).

2.2.2 Control Centre

The Control Centre is the most important unit in our framework. It is the middleware unit, the coordinator, the message handler, the "heart" of our framework. The Control Centre should normally run on a powerfull machine and it is using an IIS in order to implement the Mobile Administrator role. The Control Centre machine is using certain ports through which units get connected and send/receive XML messages. The port 80 is used from IIS for the WWW connections and two other ports are used for connecting Remote Units and Administrators. The tricky part is the Mobile Administrator connection. In this case an administrator component is instantiated on the IIS and a connection is established with the Control Centre process by using the same port that Administrative Units are using. So the Mobile Administrators are calling the IIS through port 80, the IIS is then initiating a local channel through the Administrative Units' port and the dynamically built content is returned to the Mobile Administrator. It is quite obvious that the Control Centre must separate the Remote Units and Administrators conections in order to split the functionality into different parsers and activity handlers. So we created a class that handles two communicator switches and two coordinators to serve the Remote Units and the Administrators (see Figure 6). The two cordinators (clsUnitsCoordinator and clsAdminsCoordinator) are coordinating data classes of Units (clsUnit) and Administrators (clsAdmin) respectively. The XML messages that arrive in the communicator switches are parsed in the controlling class (clsControlSrv), triggering certain activities to be executed by the coordinators. So the major components that comprise the Control Centre unit are: a Controlling Class, Communicator Switches, Coordinators, a Teleconference component and forms to present Administrators and Units.

The Units' Controlling Form that is mentioned earlier is used for presenting and handling the Remote Unit's data. We defined the XML and XSL messages which are passed by the Remote Unit in order to create an identical copy of the Remote Unit data on the Control Centre machine. These ASCII texts are received and used together in order to present a Remote Unit's XML structure in the prefered format (XSL Stylesheet). On this form a certain group of controls is also added in order to offer a set of functinalities that can change the Remote Unit's data. The same form of controller can be used from the Administrative Unit. The only difference between the two instances is that in the first one is using the coordinator for retrieving the copied unit and the second one is using directly the local copy.

The structure of the Control Centre is shown in Figure 5. The major functionality of the Control Centre is contained in the communication of the communicator switches and the controlling class. The controlling class receives a defined set of XML messages with which it is updating local copies and forwarding messages to Remote Units and Administrators. In order to define the sender of the XML messages of the switches we are using the RequestID (Socket connection) in order to uniquely determine the correct source. In our framework we have implemented all updating scenarios between the Tele-Control units and these are briefly described below:

Remote Unit is changing its XML data: The Control Centre is notified and it changes its local copy. If the copy is already requested from an Administrator unit then an update message is forwarded to that Administrator.

Control Centre is changing a Remote Unit's data on the local copy: The Control Centre notifies both the Remote Unit and the Administrator (if it exists) that is currently using a copy of the data, to update the latter.

Administrator is changing a local Remote Unit's data: The Control Centre is notified and it changes its local copy. Then the Control Centre is forwarding the same update message to the selected Remote Unit.

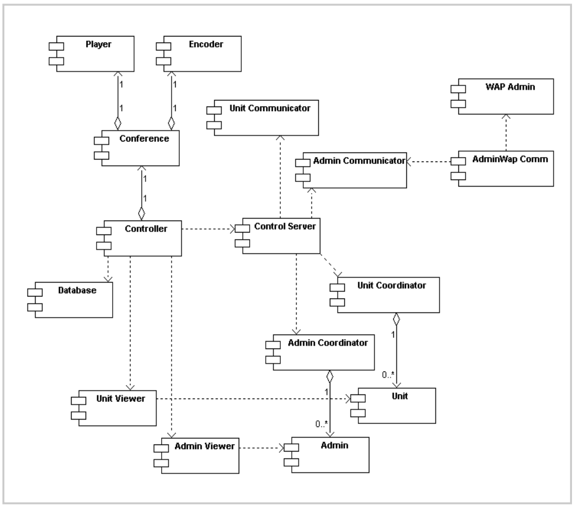


Figure 6: Administrative Unit

The other scenario that is significant to the Tele-Control System, is the Teleconference communication process. In this scenario XML messages (CONF_REQUEST, CONF_ACCEPT, CONF_CLOSE) are sent between a Remote Unit and an Administrator in order to realize a teleconference communication. The Control Centre is the middleware of the XML messages because in the teleconference communication the components are connected directly through a TCP channel.

The other scenarios that will not be described in the analysis, because they are quite simple, are:

Unit connected/disconnected from the Control Centre: It is updating the local list of connected Remote Units and then updates the list of units in all connected Administrators by a LIST message. If an Administrator is using a disconnected Remote Unit, then it is notified in order to release the copy of the Remote Unit's data.

Administrator connected/disconnected from the Control Centre: It is updating the local list of connected Administrators and if the Remote Unit that previously requested it is not released, the Control Centre releases it.

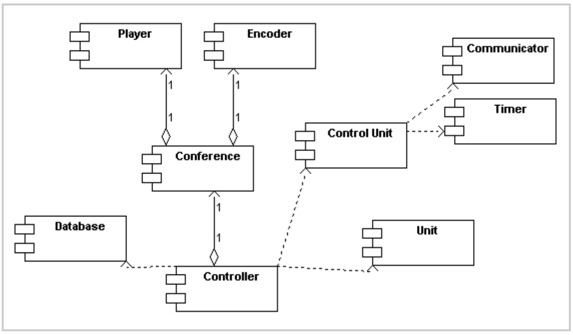


Figure 7: Administrative Unit

2.2.3 Administrative Units

The Administrative Unit structure is quite similar to the Remote Unit's one. It is using a local copy of a Remote Unit's data that is updated and informed by XML messages that are sent initialy to the Control Centre and afterwards to the original Remote Unit. The major activities that can be performed by this unit are:

- Connect/Disconnect on the Control Centre.
- Requesting a selected Remote Unit by using a list that informs the Administrator about the available units.
- Changing the local copy of a Remote Unit that is retrieved from the Control Centre. Changes are sent by using an Update Message to the Control Centre. The Control Center afterwards notifies the Remote Unit in order to update the original source.
- Initiates a Teleconference if it is needed or when it is requested from a Remote Unit.

The Administrative Unit is using a communicator to connect to the Control Centre, as in the case of the Remote Unit. The controlling form exists in the structure of this unit in order to be able to perform changes on copies of Remote Units. The administrator is using two data classes. The first one is clsAdmin that contains local information and the other one is clsUnit that contains the copy of the selected Remote Unit that is currently handled by the Administrator.

2.2.4 Mobile Administrators

The Mobile Administrators are mobile devices that can retrieve and update data of a selected Remote Unit. We have analysed the communication infrastructure of such devices, so in this section we will emphasize on the structure of the Mobile Administrators. The data in the mobile devices is dynamically built by calls on the IIS on the Control Centre machine. When a dynamic content (ASP) is requested from the Mobile Administrator, the IIS is initializing a certain component (DLL – Dynamic Linked Library) that contains the main classes of the Administrative Unit's connectivity. So we need a communicator to initialize a TCP channel to the Control Centre, a controlling class to parse the messages of the Conrol Center, and a local copy (clsUnit) of the Remote Unit's copy. The conroller is very similar to the conrolling class of the Administrative Unit but it is not triggering events; instead, it is builting WML content for mobile device. So, in general, this type of Administrator is a "virtual" Administrative Unit that comunicates with the Control Centre by using an IIS as a middleware environment. Instead of events, the ASP pages dynamically build the list of Remote Units, the "content" of a selected Unit and, on submission, they update Remote Unit data.

3 APPLICATIONS PARADIGMS

In order to prove the simplicity and reusability of our framework we used two different application paradigms. These applications are similar because their mission is the same, namely to "handle remote units". In the first case we used a Tele-Medicine system that handles pill dispenser devices and in the second one a Tele-Security system that monitors lights, doors, and alarms of a set of buildings.

3.1 Tele-Medicine application

The paradigm we used in order to define how the Remote Unit is built was a Tele-Medicine system. As already mentioned, these units handle a stack of pills stored in a dispensing machine. We will not analyse the messages that are sent within the system because the only classes and components that can be used in order to implement the Tele-Medicine system are the "Controlling Classes" and the "Teleconference Components". So the only available interface that can be used is the interface of the above class/components. During the first stages of the implementation the programmer must create the XML data that will be used from the Remote Units and the "machine" that communicates with the controlling class in order to read and write data. Here we used a COM component that contains eight stacks of pills that are also COM components. The "machine" has buttons and pills indicators that are used in order to have a GUI for our Remote Units and a Teleconference component for teleconferencing communication. The XML data that is used is shown below:

```
<TTP>
  <PrescriptionID>
    <MedicationID>
      <MedSchedules>
        <MedSchedule ID="1">
          <MedDay>Thu</MedDay>
          <MedTime>10:20:00</MedTime>
          <MedPill>1</MedPill>
          <MedNoPills>1</MedNoPills>
        </MedSchedule>  . . .
      </MedSchedules>
      <MedExpiray>10</MedExpiray>
    </MedicationID>
  </PrescriptionID>
  <Machine>
    <Pill1>
      <Active>1</Active>
      <Color>16711680</Color>
      <Count>13</Count>
      <Last>16/05/2002 11:18:17</Last>
    </Pill1> . . .
  </Machine>
</TTP>
```

The “machine” can be initialised be using the retrieving procedure of the controlling class. The interface of the machine is using events that are triggered when the patient gets a pill, the stack of pills is empty or the patient did not take his pill after a notification. In these situations we are using the SendMsg to update data on the Control Centre. The SendAlert procedure is used for sending Alert messages. Below we present a script that is used from the Remote Unit in order to initiate the stacks.

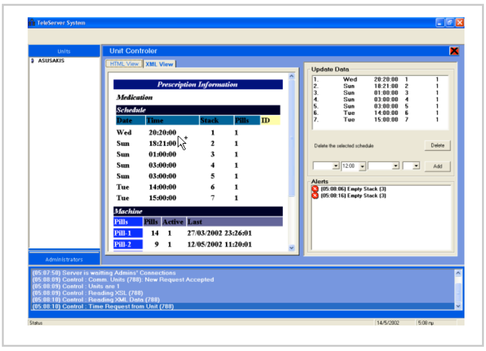
```
For i = 1 To 8
    Active(i) = cControlUnit.UnitData.
        GetElement("Machine"). _
        GetElement("Pill" & i). _
        GetElement("Active").GetValue
    Pills(i) = ControlUnit.UnitData. _
        GetElement("Machine").  _
        GetElement("Pill" & i). _
        GetElement("Count").GetValue
    Color(i) = ControlUnit.UnitData. _
        GetElement("Machine"). _
        GetElement("Pill" & i). _
        GetElement("Color").GetValue
    LastTaken(i) =
        ControlUnit.UnitData.
        GetElement("Machine"). _
        GetElement("Pill" & i). _
        GetElement("Last").GetValue
Next i
For i = 1 To 8
    Call Me.Machine.InitMachine(i, _
        CLng(Color(i)), _
        CInt(Pills(i)),
        CBool(Active(i)),
        CDate(LastTaken(i)))
Next i
```

The following script is used when a patient gets a pill from a stack. A pill is taken according to a predefined medication schedule.

```
cControlUnit.UnitData. _
    GetElement("Machine", strPath). _
    GetElement("Pill"&index, strPath).
    GetElement("Count").SetValue
    Machine.PillslCount(index) - 1
cControlUnit.UnitData. _
    GetElement("Machine"). _
    GetElement("Pill" & index). _ _
    GetElement("Last").SetValue
    Format(Now, "dd/mm/yyyy hh:mm:ss")
strXML = cControlUnit.UnitData. _
    GetElement("Machine").
    GetElement("Pill" & index).XML
Call cControlUnit. _
    UpdateTTP(strPath, strXML)
```

The scenarios mentioned above dynamically conform to the new Remote Unit type. The only change in the Control Centre and the Administrative Units is the “Controlling Form” that is used in order to update the Remote Units scheduling. The medication is the only information in the Remote Units XML structure that can be changed, so in the form we used a list of current schedules and controls to add new schedules in the list. When additions and removals take place, XML messages are constructed and sent to other Tele-Control units. In Figure 8 we show how the Tele-Medicine Control Centre looks like, and in particular: a) the list of connected Units on the Control Centre, b) the XML/XSL information of the selected Remote Unit in a Web Browser control, c) the controls that are used for listing, deleting, adding new schedules (on the right). When a new schedule is added in the Remote Unit copy, the update scenarios are triggered in order to inform the original source and other copies to change.

The Administrative Units are using the same GUI and the same “Controlling Form” in order to update the Remote Units. In the Mobile Administrators the ASP pages that are used, retrieve dynamic content from Control Centre and they are presenting the WML on the mobile device. The above environment is fully functional and the framework we implemented was successfully used in the Tele-Medicine application.



Prescription Information

Medication

Schedule

Date	Time	Stack	Pills	ID
Wed	20:20:00	1	1	
Sun	18:21:00	2	1	
Sun	01:00:00	3	1	
Sun	03:00:00	4	1	
Sun	03:00:00	5	1	
Tue	14:00:00	6	1	
Tue	15:00:00	7	1	

Machine

Pills	Pills	Active	Last
Pill-1	14	1	27/03/2002 23:26:01
Pill-2	9	1	12/05/2002 11:20:01

```
1.   Wed   20:20:00   1   1
2.   Sun   18:21:00   2   1
3.   Sun   01:00:00   3   1
4.   Sun   03:00:00   4   1
5.   Sun   03:00:00   5   1
6.   Tue   14:00:00   6   1
7.   Tue   15:00:00   7   1
```

```
(05:08:06) Empty Stack (3)
(05:08:16) Empty Stack (3)
```

```
(05:07:50) Server is waitting Admins' Connections
(05:08:09) Control : Comm. Units (788): New Request Accepted
(05:08:09) Control : Units are 1
(05:08:09) Control : Reading XSL (788)
(05:08:10) Control : Reading XML Data (788)
(05:08:10) Control : Time Request from Unit (788)
```

Figure 8: Tele-Medicine machine

3.2 Tele-Security application

The second paradigm is a security system that handles lights, doors and alarms in a set of buildings. The implementation of this system is quite similar to the previous application. We used an XML structure to contain information about a building and then a “machine” component that handles the external devices and sensors (a simulator component). In the “machine’s” GUI we are using components that simulates rooms and doors, and an XML structure that holds information about their status. In the same way the messages of the Tele-Security system are forwarded to the communicating units and Remote Units’ information is passed as a copy and gets updated by the Administrators.

4 RELATED WORK

Distributed applications are based on the distribution of components through communication channels. Many of them are using connectors and well-defined interfaces to put into operation the distributed systems (Bruneton, 2001). The connectors are constructed on middleware, mainly CORBA and DCOM. Although these platforms provide various methods, they do not by themselves provide a simple messaging binding between the nodes. Additionally, there is some serious research in implementing middleware that is using XML documents between the connected nodes (Lowe, 2002) which is much simpler. The XML middleware (Ciancarini, 2002) can implement complex applications based on XML Document agents. In our work we are using the XML serialisation technique not only to transmit data object but also to exchange coordination and control messages. The recent MOM (Message-Oriented Middleware) Servers are extending the standard XML into a powerful set of instructions and procedure calls, executed on the distributed nodes (Cabri, 2000). In our work we are implementing a specific framework that is using the above range of techniques to specify a well-defined pattern of Tele-Control Systems. It provides not only the communication techniques between the nodes but also the coordination signaling that is needed to handle remote administrators and remote units.

5 CONCLUSIONS

Generic frameworks comprising reusable components are currently becoming more popular because they can assist in the rapid development of large-scale and complicated applications. The Tele-Control framework has already proved that we can easily create an application that handles Remote Units. The developer can use this framework without any knowledge of sockets or XML messages. We believe that the XML protocol that we are using covers the major scenarios of the communication process. The dynamic structures of the XML parsing classes verify that the controlling classes of the framework can handle any formed XML structure.

REFERENCES

Clemens Szyperski, 1998. *Component Software, Beyond Object-Oriented Programming,* Addison-Wesley.

Fred Halsall, 1996. *Data Communications, Computer Networks and Open Systems*, Addison-Wesley.

Eric Bruneton, Michel Riveill, 2002. An architecture for extensible middleware platforms, *IASTED 2002*.

Welf Lowe, Markus L. Noga, 2002. A Lightweight XML-based Middleware Architecture, *20th IASTED International Multi-Conference Applied Informatics.*

Giacomo Cabri, Letizia Leonardi, Franco Zambonelli, 2000. XML Dataspaces for Mobile Agent Coordination, *SAC 2000.*

Paolo Ciancarini, Robert Tolksdorf, 2002. Coordination Middleware for XML-centric Applications, *SAC 2002.*